Community Detection for Large Graphs
Chengbin Peng1, Tamara G. Kolda2, Ali Pinar2, Zhihua Zhang3, David E. Keyes1

1 CEMSE, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
2 Sandia National Laboratories, Livermore, CA, USA
3 Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China

Abstract
Many real world networks have inherent community structures, including social networks, transportation networks, biological networks, etc. For large scale networks with millions or billions of nodes in real-world applications, accelerating current community detection algorithms is in demand, and we present two approaches to tackle this issue:
• a K-core based framework that can accelerate existing community detection algorithms significantly;
• a parallel inference algorithm via stochastic block models that can distribute the workload.

Introduction
The community structure of a graph reflects the connectivity between different nodes. Within each community, nodes are more likely to be connected to each other than to nodes from different communities. If each node is allowed to exist in only one community, the community structure is called non-overlapping. In this work, we consider the non-overlapping case only. Recently, algorithms for large-scale community detection have been devised, with linear or near-linear time complexity. However, even those linear-time algorithms become slow when detecting community structures, including social networks, transportation networks, biological networks, etc. For many real world networks have inherent community structures.

K-core Based Acceleration Framework
The K-core of a graph is the largest subgraph within which each node has at least K connections.

Advantages
• Applicable to most of the existing algorithms;
• Fast and preserving the quality (in terms of modularity).

Framework: 3-Step Algorithm
1. $G_K \leftarrow \text{Kcore}(G, K)$
2. $g_k \leftarrow \text{CommunityDetection}(G_K)$
3. $g \leftarrow \text{Recover}(G, g_k)$

Here, G is the original graph, g is a vector of community labels, K is some integer, $G_K$ is the K-core, and $g_k$ contains community labels for the K-core.

Why does it work?
• Intuition: Groups of nodes with denser connections (intra-community connections) will still have dense connection in the K-core
• Theorem: If the algorithm is to maximize the likelihood function, the community labels of the remaining nodes given $G_K$ is expected to be the same as when given G.

Parallel Inferenceing Algorithm
Stochastic Block Models (SBM) are commonly used for community detection.

SBM Model
Each node belongs to one of the m blocks and we use $Z \in \{0,1\}^{n \times m}$ to represent the block labels. $Z_{ik} = 1$ means node $i$ belongs to block $k$ and in each row of $Z$ only one entry can be one. We also define a matrix $B \in \{0, 1\}^{m \times m}$ where $B_{ik} = 1$ ($k_1 \neq k_2$) represents the connecting probability between two nodes from block $k_1$ and $k_2$ respectively. If $k_1 = k_2$, $B_{k_1k_1}$ represents the connecting probability inside the block.

Using the matrices B and Z, we define a probability matrix $\Theta = ZBZ^T$. Then the adjacency matrix G of a sample network can be drawn from the model by

$$\Pr(G_{ij}) = \begin{cases} \Theta_{ij} & \text{if } G_{ij} = 1, \\ 1 - \Theta_{ij} & \text{if } G_{ij} = 0, \end{cases}$$

for $i, j \in \{1, 2, \ldots, n\}$ and $i \neq j$. Typically, the adjacency matrix G is available from the data set. Our main purpose is to infer Z.

If only G is given, the log-likelihood function is

$$L(B, Z|G) = \frac{1}{n} \sum_{i,j} \log \Pr(G_{ij}) = \frac{1}{n} \sum_{i,j} \log [(1 - G_{ij}) + (2G_{ij} - 1)\Theta_{ij}].$$

It is difficult to maximize such a likelihood function through traditional optimization methods, and consequently to find the optimal Z and B.

Algorithm
• Approximate B using its diagonal vector;
• Partition G, Z and B respectively to all the processors;
• Alternatively update $Z \leftarrow \text{argmax}_Z L(B, Z|G)$ and $B \leftarrow \text{argmax}_B L(B, Z|G)$ until converge
  • Run in parallel
  • When updating Z, only communicate the change of connected nodes in the network
  • When updating B, only communicate the changed blocks.

The computing time is a linear function of the number of edges, and the communication cost grows linearly to the number of parallel processes.

Experiments

Contact Information
• Email: Chengbin.Peng@kaust.edu.sa

References